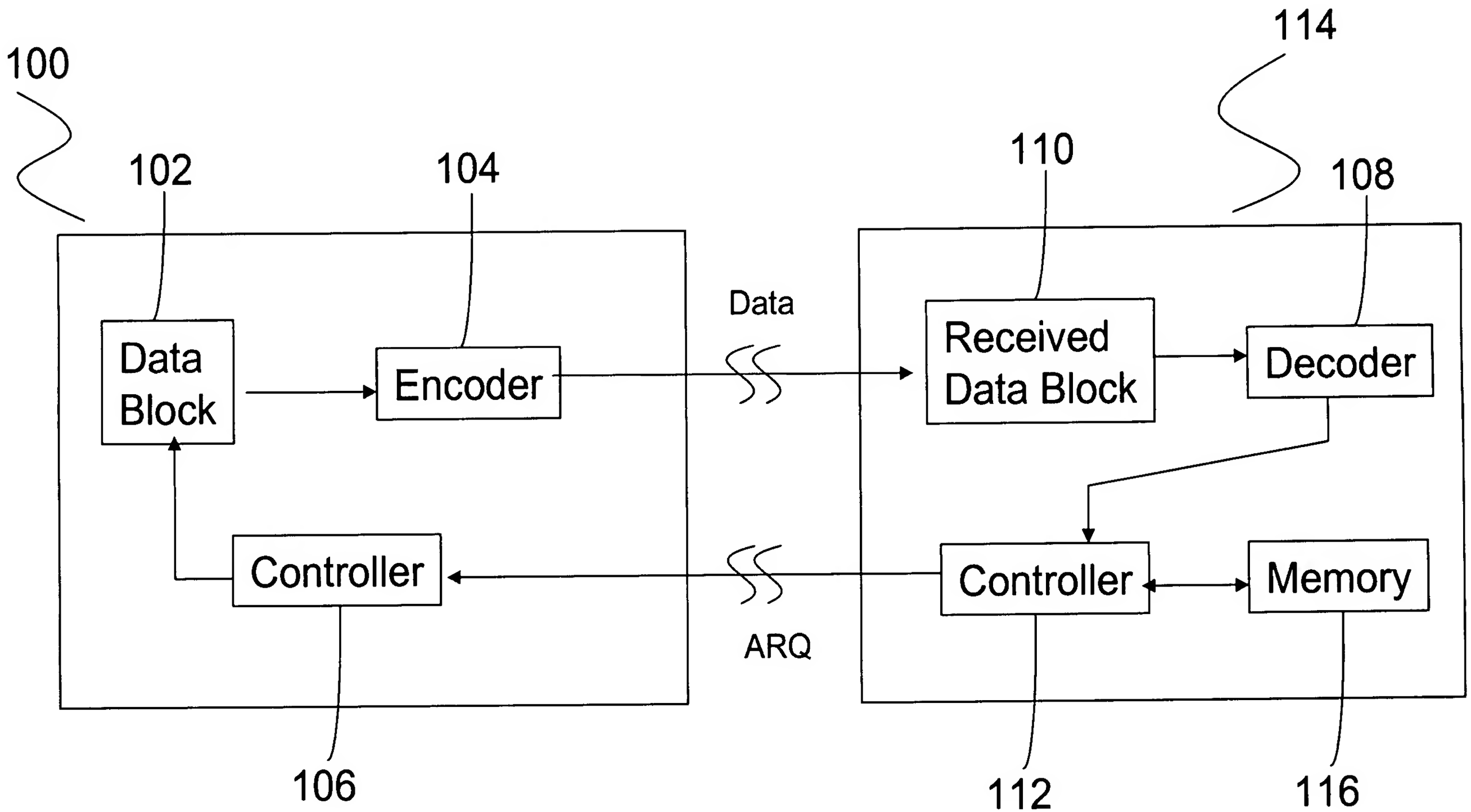
100
102
104
110
114
108
Data Block
Encoder
Data
Received Data Block
Decoder
Controller
Controller
Memory
ARQ
106
112
116

FIG. 1

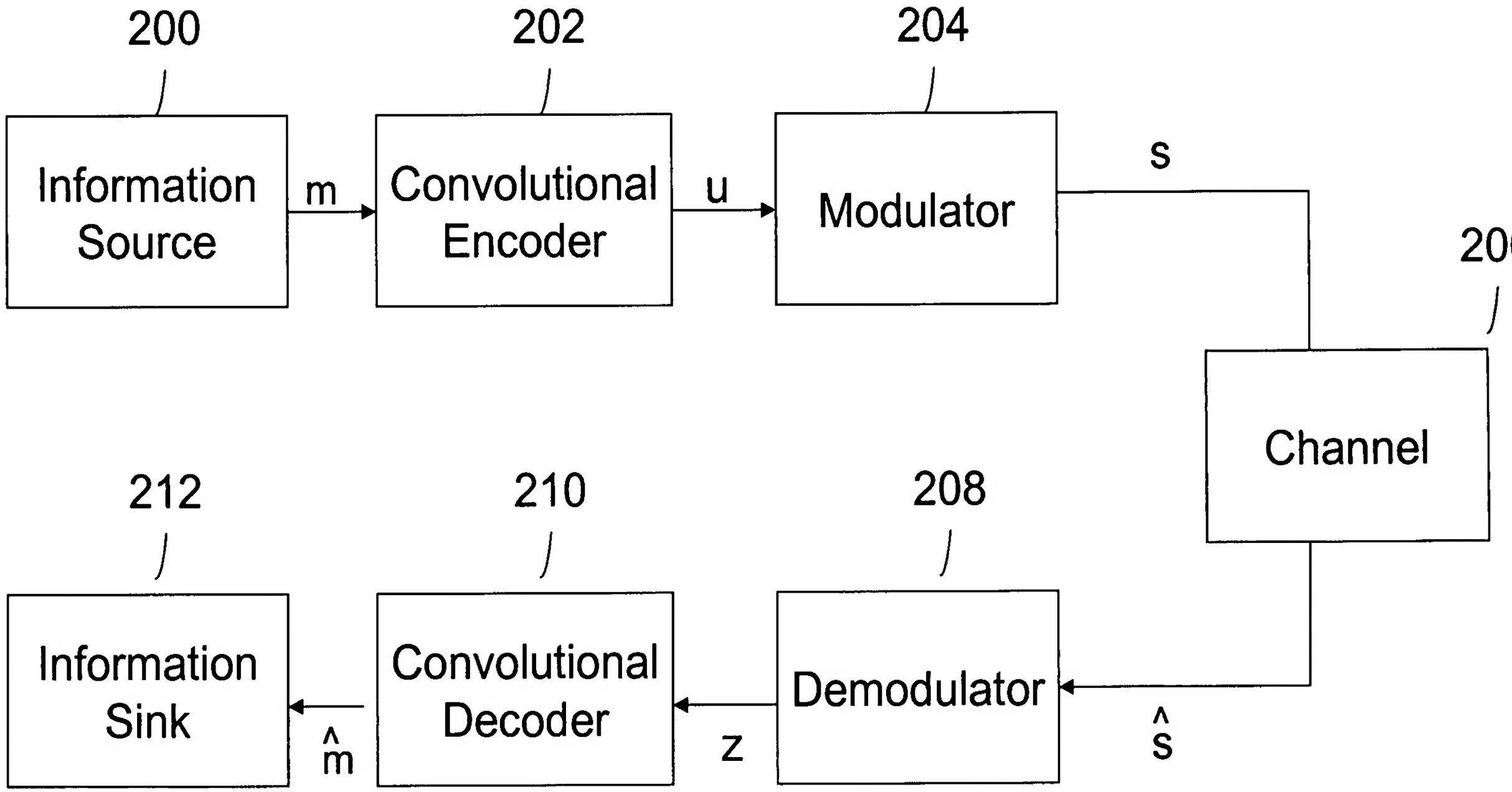
200
202
204
Information Source
m
Convolutional Encoder
u
Modulator
s
206
Channel
212
210
208
Information Sink
m̂
Convolutional Decoder
z
Demodulator
ŝ

FIG. 2

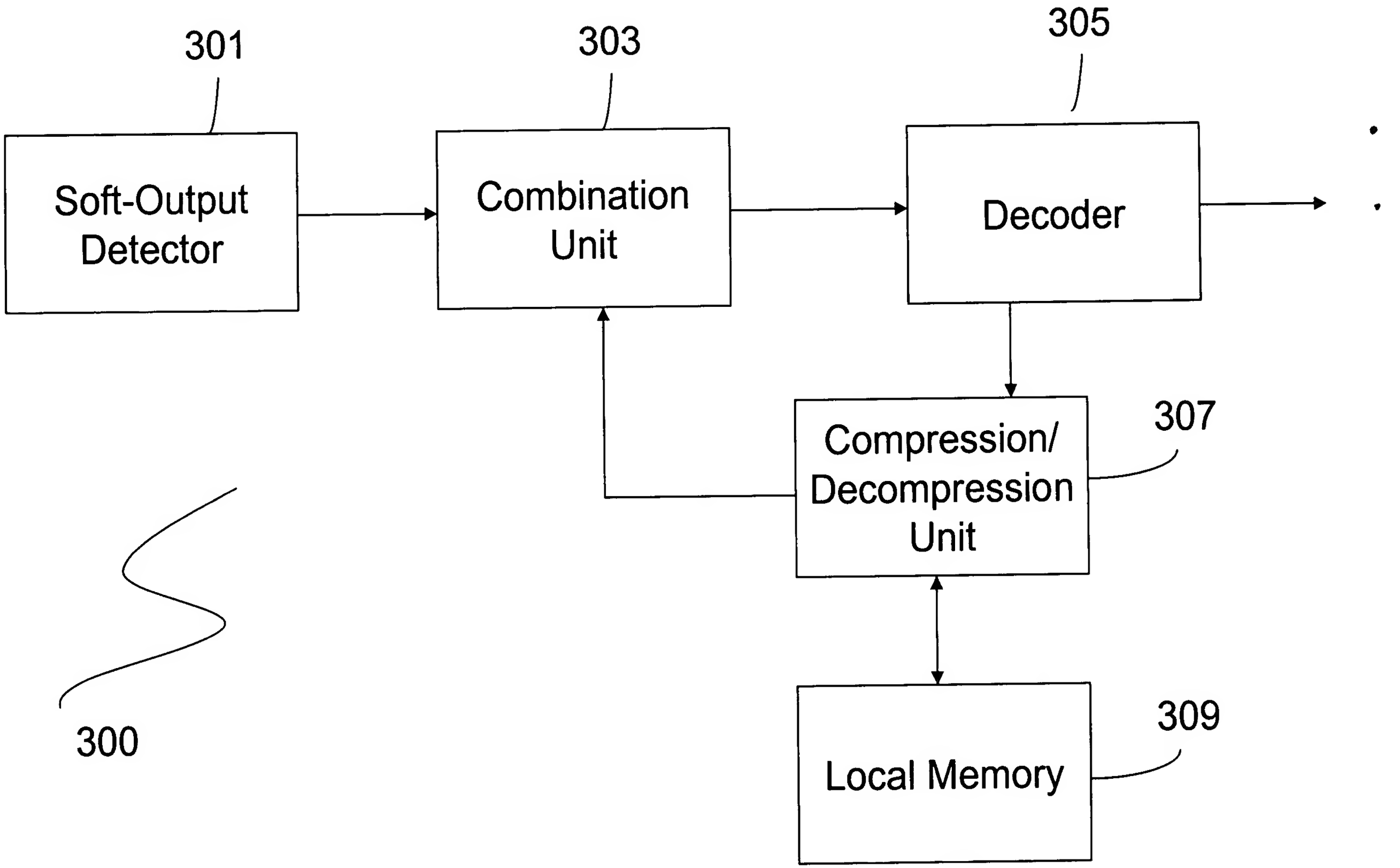
301
303
305
Soft-Output Detector
Combination Unit
Decoder
Compression/ Decompression Unit
307
Local Memory
309
300

FIG. 3

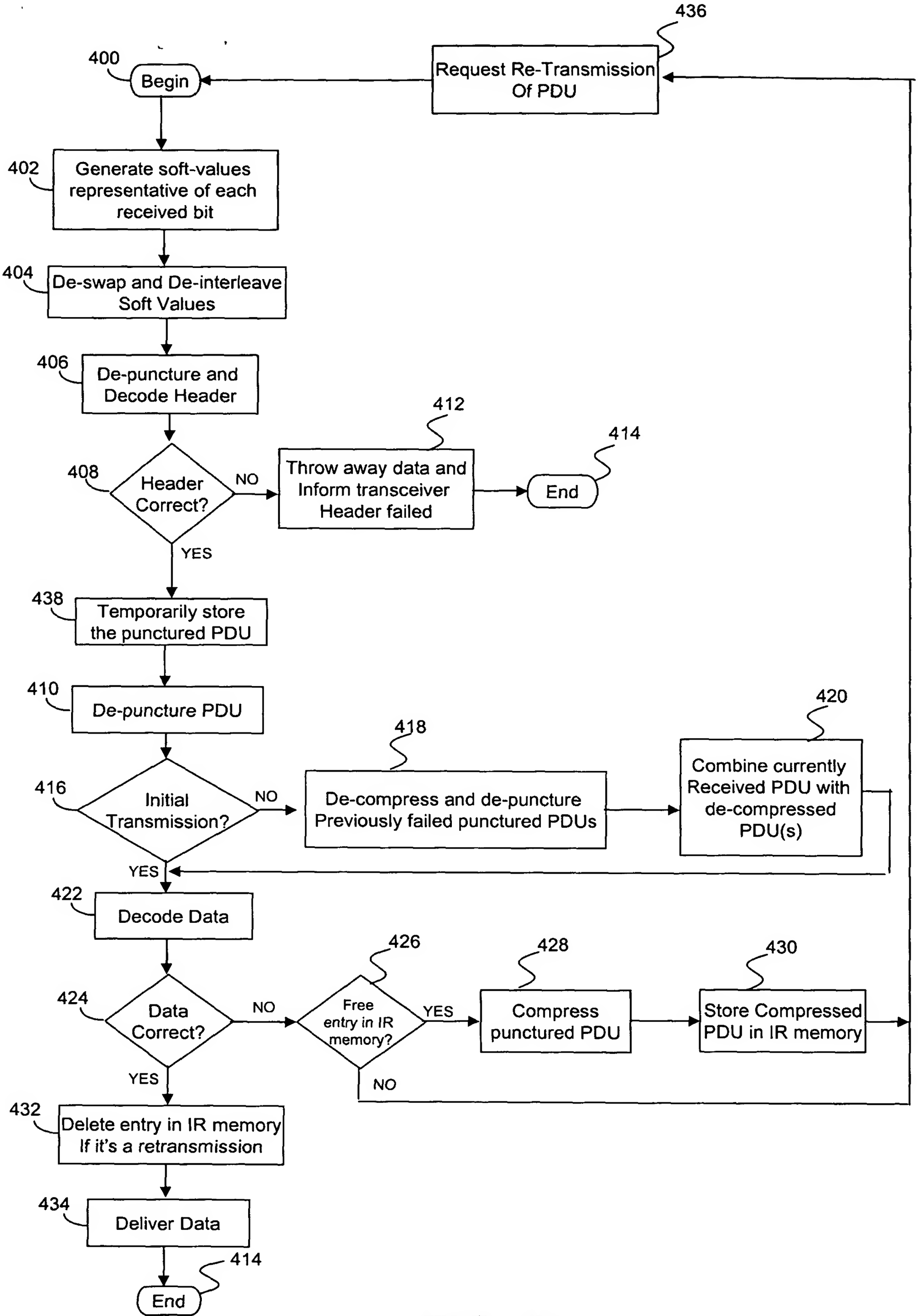
400
Begin
436
Request Re-Transmission Of PDU
402
Generate soft-values representative of each received bit
404
De-swap and De-interleave Soft Values
406
De-puncture and Decode Header
412
414
408
Header Correct?
NO
Throw away data and Inform transceiver Header failed
End
YES
438
Temporarily store the punctured PDU
410
De-puncture PDU
420
418
416
Initial Transmission?
NO
De-compress and de-puncture Previously failed punctured PDUs
Combine currently Received PDU with de-compressed PDU(s)
YES
422
Decode Data
426
428
430
424
Data Correct?
NO
Free entry in IR memory?
YES
Compress punctured PDU
Store Compressed PDU in IR memory
YES
NO
432
Delete entry in IR memory If it's a retransmission
434
Deliver Data
414
End

FIG. 4a

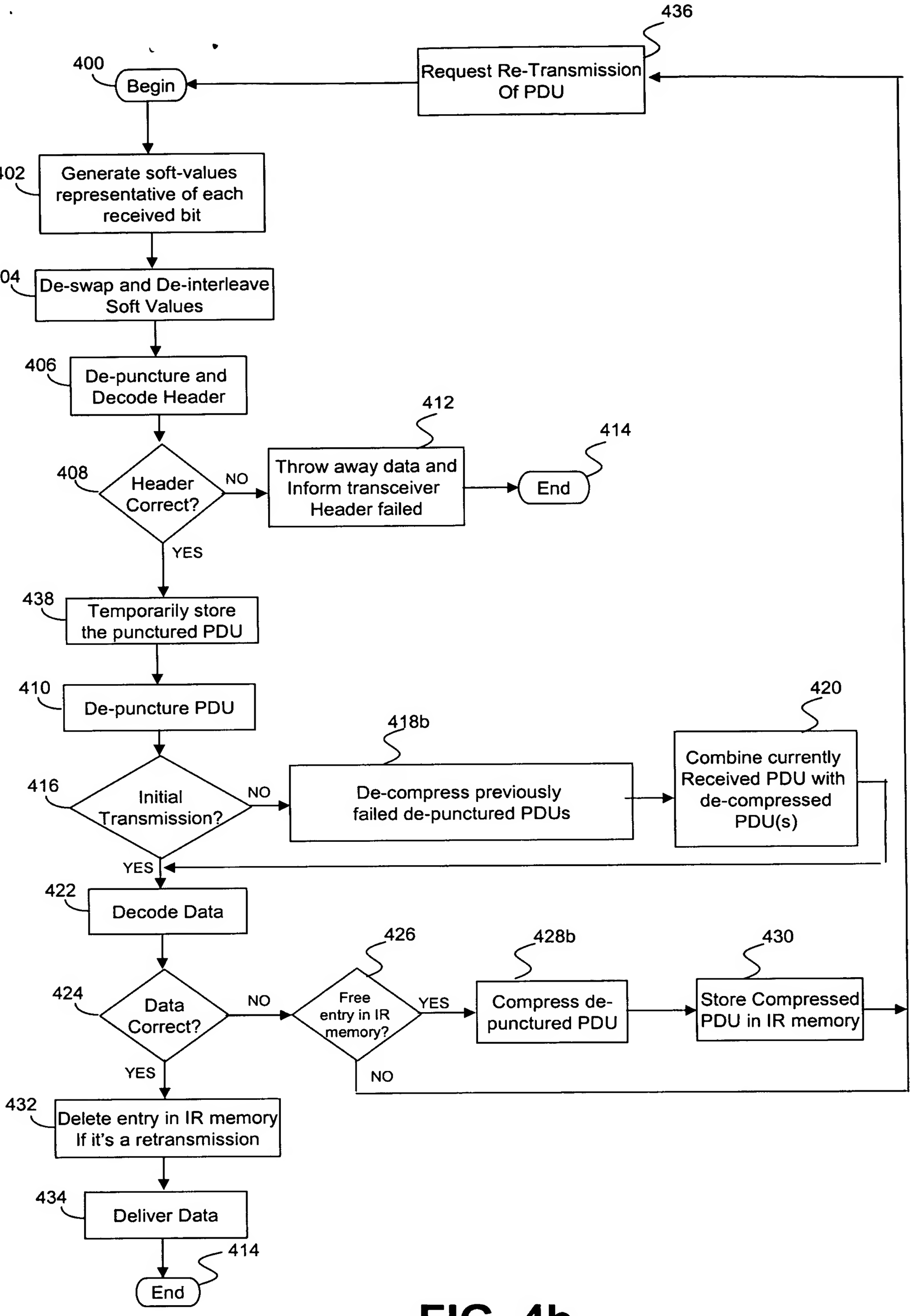
400
Begin
402
Generate soft-values representative of each received bit
404
De-swap and De-interleave Soft Values
406
De-puncture and Decode Header
408
Header Correct?
NO
412
Throw away data and Inform transceiver Header failed
414
End
YES
436
Request Re-Transmission Of PDU
438
Temporarily store the punctured PDU
410
De-puncture PDU
416
Initial Transmission?
NO
418b
De-compress previously failed de-punctured PDUs
420
Combine currently Received PDU with de-compressed PDU(s)
YES
422
Decode Data
424
Data Correct?
NO
426
Free entry in IR memory?
YES
428b
Compress de-punctured PDU
430
Store Compressed PDU in IR memory
YES
NO
432
Delete entry in IR memory If it's a retransmission
434
Deliver Data
414
End

FIG. 4b

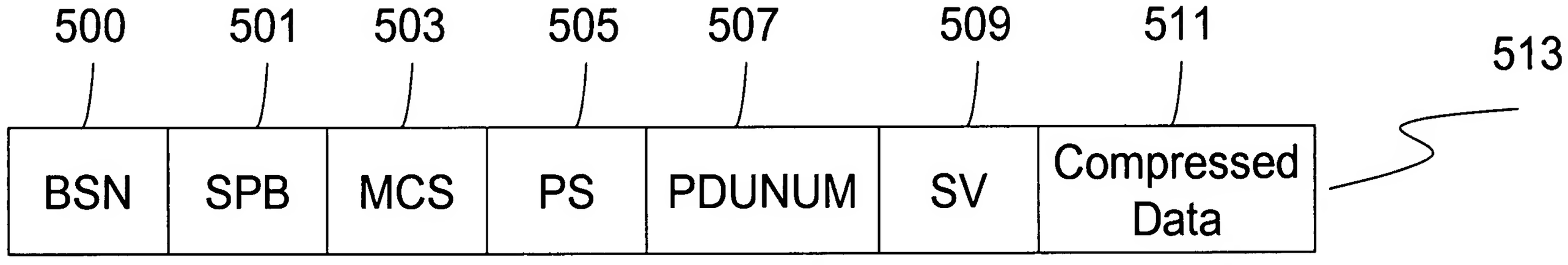

FIG. 5

BSN	SPB	MCS	PS	PDUNUM	SV	Compressed Data
88	0	9	1	1	$\bar{\alpha}$	Compressed Data 1
88	0	9	2	1	$\bar{\beta}$	Compressed Data 2
88	0	9	3	1	$\bar{\gamma}$	Compressed Data 3

FIG. 6

```
for burstNum = 1 to 4
    Sum = 0;
    for pos = 1 to BurstLength
        Sum = sum + receivedVector[burstNum, pos];
    end //pos
    ScaleVector[burstNum] = sum/BurstLength
End //burst
```

FIG. 7A

```
for pos = 1 to pduLength
    if puncturedVector[pos] < 0
        storedVector[pos] = 1;
    else
        storedVector[pos] = 0;
end //pos
```

FIG. 7B

```
For pos = 1 to pduLength
    decompressedVector[pos] = storedVector[pos] * scaleVector[scaleIndexTable[pos]];
End //pos
```

FIG. 7C

```
For pos=1 to unpuncturedPduLength
    if (scaleindexTableUnpunctured[pos] == 0)
       decompressedVector[pos] = o
    else
       decompressedVector[pos] = storedVector[pos] * scaleIndexTableUnpunctured[pos]
    end
end
```

FIG. 7D

```
combinedVector[ ] = newDepuncturedVector[ ];
for I = 1 to numIREntries
     if (IREntry[i] -> BSN = = currentBSN and
                IREntry[i]->SPB == currentSPB)
                          decompressedVector[ ] = decompress(IREntry[i]);
                          depuncturedVector[ ] = depuncture(decompressedVector[ ],
                                                IREntry[I].MSC);
                          combinedVector[ ] = combinedVector[ ] + depuncturedVector[ ];
    end //if
end //
```

FIG. 7E